Technische Universität Berlin

FAKULTÄT II
MATHEMATIK UND
NATURWISSENSCHAFTEN
Institut für Mathematik

Delay Resistant Timetabling

by

Christian Liebchen and Sebastian Stiller

No. 2006/24

Delay Resistant Timetabling

Christian Liebchen and Sebastian Stiller*

23rd November 2006

In public transport punctuality has prominent influence on the customers' satisfaction. Our task is to support a management decision to optimally invest passengers' nominal travel time to secure the nominal schedule against delay. For aperiodic scheduling we clarify the notion and use of a fixed amount of time supplements, so-called buffers, both theoretically and by realistic examples. The general tool to solve such optimization problems is a sampling approach. We show how this approach is mathematically justified. As its applicability to large networks is limited, we show an efficient alternative for the case of series-parallel graphs. For periodic timetabling we propose two heuristic approaches to ensure a certain level of delay resistance at the least expense of passengers travel time, and analyze in detail their advantages and drawbacks.

1 Introduction

In public transport punctuality has prominent influence on the customers' satisfaction. A train that is five minutes late may cause much more distress than a train that is scheduled to run five minutes longer.

Besides technical means the design of the schedule is widely expected to contain potential for reducing delays. This can be used in an obvious way, namely by scheduling all trains so loosely that virtually every train in any scenario will be on time. Of course, this is heavily at the expense of the customers' travel time. And these are of major importance for the choice of the means of transport. Hence, one should strive for a balance between the *delay resistance* of a schedule and the *price* that is incurred for this delay resistance. A very global such policy is recommended by the International Union of Railways (UIC): Instead of using the technical minimal time durations for the running times, every schedule should respect a buffer of 7% of these technical minimal times.

Related work. One may think of more sophisticated policies. Similar problems have been solved by techniques of robust optimization ([1]), e.g. in supply chain management. However, robust approaches are hard to apply to these prob-

*This work has been supported by the DFG Research Center Matheon in Berlin, and by the ARRIVAL project, within the 6th Framework Programme of the European Comission under contract no. FP6-021235-2.

lems. Technically speaking the matrix mathematically encoding these problems is so sparse that robust optimization leads to over conservative solutions.

Moreover, applying such techniques to timetabling in public transport encounters an impedient asymmetry, which we call the *timetable condition*: By defining a schedule, for every single trip of a vehicle between two consecutive stops the transportation company gives the following service guarantees

- The trip *must* not depart prior to the published departure time.
- The trip *should* not arrive later than the published arrival time.

Hereby, delays can only be absorbed by time supplements at later activities. In particular, it is not feasible to prevent delays by operating a trip earlier than it has been scheduled. Thereby the information contained in the expectation is devaluated. The timetabling condition reduces both the optimization potential and the applicable tool box of optimization methods.

Recently, Kroon et al. [4] addressed several questions that arise in this context. For instance, they consider a trip between two terminal stations. By means of random sampling they distribute a given budget of buffer times over the segments between two consecutive stops. Their computations suggest that applying the global UIC policy in general is only suboptimal. Rather, strictly more than half of the global UIC budget of buffer times should be located strictly before the middle number of stations of that trip.

Contribution. We consider a setting that is much similar to the one of Kroon et al. [4]. Our analysis provides a theoretical explanation why buffer times should be distributed in such an asymmetric way. Kroon et al. prove the convergence for a path of length 2. We prove convergence and convergence speed for arbitrary directed acyclic graphs and show the convexity of the corresponding programs. To this end we first have to sharpen the notion of a buffer budget for arbitrary directed acyclic graphs which highlights a relevant anomaly in the UIC rules.

Finally, we also address explicitly the construction of periodic timetables, because most European railway companies are operating periodic timetables. There, the Periodic Event Scheduling Problem (PESP) is widely used as the model of choice, see [6, 5] for its modeling features and several Integer Programming (IP) formulations. We propose two ways for incorporating a certain degree of robustness into the corresponding IP—with only a moderate loss in nominal quality.

The final goal is to support a management decision on how much one is willing to pay for delay resistance, and to ensure, that for a certain budget of buffer time the maximum resistance against delays is achieved.

2 The aperiodic case

2.1 How to account for the price of delay resistance

In the aperiodic case we are confronted with a DAG with stochastic arc length, of which we know a lower bound, i.e., the technical travel time, t_a. The aim is

to distribute a fixed budget of time as supplements s_a, so-called buffers, such that an earliest start schedule respecting $t_a + s_a$ has minimal expected delay.

The key problem of this concept is, that by placing a buffer in a graph with a topology other than a tree may cause indirect buffering at other arcs. Consider two in-going arcs, a and b, of a node v that would—if unbuffered—allow for the same start time at v. Any placed buffer on a will imply buffering b, too. Thus, if we only count the buffers placed, we will have an unfair accounting against the budget. We will investigate this phenomenon in detail. First, we consider only those graphs where it does not occur, namely trees.

2.1.1 Trees

It is reasonable to assume the distributions of disturbances on each arc as discretized and finite, in particular bounded. This allows to account for expectation as finite weighted sums over a set of scenarios. This set may in total still be far too big for a computational purpose. Consider the example of a bus trip with 20 bus stops. Each section from one stop to the next independently experiences a fixed disturbance of δ with probability p, i.e., the i-th section takes t_i time units with probability $1 - p$ and $t_i + \delta$ in the other cases. Each section or arc has only two values, but the whole system has about 10^{30} different scenarios. The requirement that the number of scenarios is finite can be dropped for some of the results.

For our model let $T = (V, A)$ be a directed graph of which the underlying undirected graph is a tree $t : A(T) \to \mathbb{N}^+, a \mapsto t_a$ the function of the minimal durations of an arc, R the set of scenarios endowed with the discrete sigma algebra, and a probability distribution p for the scenarios, $\delta_a : R \to \mathbb{N}^+, r \mapsto \delta_a^r \ \forall a \in A(T)$ a family of random variables expressing the seminal *disturbance* on arc a in scenario r.

Definition 1. A function $s : A(T) \to \mathbb{Q}, a \mapsto s_a$ is a *buffering* of T, and called *feasible* for budget $B \in \mathbb{Q}$, if and only if $\sum_{a \in A(T)} s_a \leq B$.

In the simplest case the weight function equals the number of passengers, who get off the train at the end of an arc. Note that we do not require a buffering to be non-negative. Obviously, the set of all feasible bufferings for a budget B is convex. We abbreviate the arc set of T as $A := A(T)$.

Given T, t and a buffering s of T for budget B and some passenger load related weight ω_a, we can calculate a nominal schedule x that minimizes the total travel time of passengers and respects $t_a + s_a$ as the minimum difference between the start- and the end-vertex of an arc a. The time assigned to the end-vertex of an arc a in this schedule is x_a.

Moreover, one can use a second weight function g_a on the arcs related to the number of passengers leaving the system after that arc. Whereas ω is used to calculate the total weighted travel time, the total delay shall be weighted by g. A passengers enters the system at the starting time of his first trip in the nominal schedule. He leaves the system, when his last arrival event happens

according to the dispatched, actual schedule. The delay he experiences is the delay of the last arc he used.

Given $T, t, (R, p), (\delta_a)_A$ and a buffering s of T for budget B, we can for each scenario r calculate a (dispatching) schedule x^r that minimizes the total travel time of passengers under the following two requirements: First, the schedule respects $t_a + \delta_a$ as the minimum difference between the start- and the end-vertex of an arc a. Second, the time assigned to the end-vertex of an arc a in the schedule for scenario r is greater or equal to x_a. Denote the corresponding random variable for the time of the end-vertex of an arc a as x_a^r.

Definition 2. The random variables $d_a : R \to \mathbb{Q}^+, r \mapsto d_a^r, \forall a \in A$ defined by $d_a^r := x_a^r - x_a$ are called the *delay* on arc a. The expectation of the random variable $g^\mathsf{T} d = \sum_{a \in A} g_a d_a$ is called the (total, weighted) expected delay.

Sometimes we will use the vertex itself as the index of a scheduled time, e.g., for $a = (u, v), b = (w, v) \in A(T)$ we have $x_a = x_b = x_v$ by definition.

As the underlying graph T of precedence constraints is a tree, the optimal nominal and dispatching schedules are easy to find and unique up to the time of one (initial) vertex. In particular, the foregoing notions are well-defined. One creates the values x_a by starting at some vertex and propagating the time through the tree such that for each arc $a = (v, w)$ the requirement $x_w - x_v \leq t_a + s_a$ are fulfilled with equality. Note that this propagation also follows backward arcs. For the schedule of a scenario use some topological order of T. Iterating along this order, the time for each vertex v is set to the maximum of the previous scheduled time x_v and all $x_a^r + t_a + \delta_a^r, \forall a = (u, v)$.

In fact, on a tree the technical travel time t is immaterial. Because the nominal schedule x is tight at every vertex, we can consistently define for every vertex v the delay in scenario r as $d_v^r = d_a^r, \forall a = (u, v) \in A(T)$. This yields $d_v^r = \max\{0, \max\{d_u^r + \delta_a^r - s_a | a = (u, v) \in A(T)\}\}$, which can be calculated directly along some topological order. In this description we can immediately model the UIC's [10, 4] rule of adding 7% to each trip's travel time, i.e., $s_a = 0.07 \cdot t_a$. This buffering is feasible for any budget greater or equal to $0.07 \cdot \sum_a t_a$.

Theorem 1. For given $T, \omega, g, (R, p), (\delta_a)_A$ and budget B the function f that maps every feasible buffering s on the resulting expected, total, weighted delay $E[g^\mathsf{T} d^r]$ is *convex*.

Proof. We show that for a fixed scenario r and a fixed arc a the delay d_a^r is a convex function of the $|A|$-dimensional vector s. Recall that $d_{a=(u,v)}^r = d_v^r = \max\{0, \max\{d_u^r + \delta_a^r - s_a | a = (u, v) \in A(T)\}\}$. Pick some topological order and argue by induction along it. To start the induction observe that the delay of a vertex without predecessors is constantly 0 and thus convex. Therefore d_a^r is the maximum of a finite family of—by induction—convex functions, and therefore convex itself.

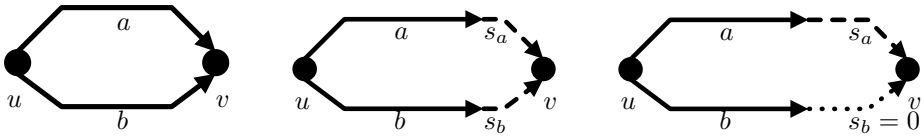
This gives that f is a weighted sum (expectation) of weighted sums (summing over all arcs) of convex functions, thus f itself is convex. □

2.1.2 Non-tree topologies

In the section on trees we gave a notion of buffers that corresponds well to the UIC rule. A buffer is a time supplement on an *arc*. The nominal schedules x, the timetables, which arose from such a buffering were tight at every vertex. The delay of a realized schedule in a certain scenario against the nominal schedule is thus the same for all arcs entering a certain vertex. In this way we could for the delays switch from the arc- to the vertex-perspective. Still, the buffers are understood as supplements on arcs.

This proves to be misleading in the case of underlying networks that do not feature a tree topology. Theoretically speaking, the objective function f, the expected total weighted delay is no longer convex. In practice this amounts to a perfectly unfair accounting of the price that is paid for delay resistance.

We will now investigate both the theoretical and the practical consequences of the matter and then present a fair accounting of the price of resistance, that is actually proven to measure the nominal, total, weighted prolongation of passengers' travel time. In this section we start with the same notation as in the previous, except that T is replaced by G symbolizing an arbitrary directed, acyclic graph.



(a) A simple non-tree example

(b) The UIC buffering

(c) An optimal, unfair buffering

Figure 1: Optimization leads to unfair buffering

Consider the network in Figure 2.1.2. Let the technical travel times $t_a = t_b = 100$ be equal. The UIC's rule will assign buffers of $s_a = s_b = 7$ time units to both arcs, cf. Figure 2.1.2. For $x_u = 0$ this yields $x_v = 107$. Hence, a budget $B = 14$ is in tune with the UIC's rule. But, any optimizing procedure that is allowed to choose among all bufferings constraint to budget B will put all 14 time units of supplement on one of the arcs, say a, and not buffer the other arc at all. This will set the time difference from x_u to x_v to 114. Any nominal schedule that respects the buffered travel time on arc a has to create a huge, indirect buffer (the dotted line) on arc b in Figure 2.1.2. In some sense the arc b is a free-rider. In every nominal schedule it will have a buffer of at least 14 time units, but if it comes to the accounting against the budget B its buffer s_b counts 0. Obviously, the later is the optimal, though unfair buffering. It is also obvious, that this buffering has a much higher nominal, total travel time. That is why we call it unfair.

One might object, that the passenger on arc b may already get off earlier. But the example network can be part of a greater network as in Figure 2. If most passengers travel from vertex A or vertex B beyond vertex v we encounter

the same problems as above.

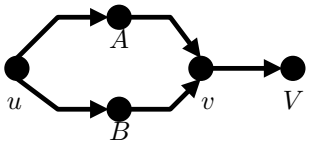


Figure 2: Passengers mainly travel from A and B to V

In the next section we present a detailed real-world example with the additional feature of a *periodic* timetable.

Consider again Figure 1 together with a set $R = \{1, 2\}$ of two scenarios. Set $\delta_a^1 = 10, \delta_b^2 = 10$ and the two other disturbances to 0. The weight function shall be trivial $w \equiv 1$. The budget is again $B = 14$. For the two buffering $s_a = 14, s_b = 0$ and $s'_a = 0, s'_b = 14$ the value of the objective function $f(s) = f(s')$ is 0. Yet, their convex combination for $s^\lambda = \lambda s + (1 - \lambda)s'$ for $\lambda = \frac{1}{2}$ yields $f(s^{\frac{1}{2}}) = 6$, thus, contradicting the convexity.

Here the absence of the convexity is the formal expression of the unfair accounting. The key problem is, that by placing a buffer in a graph with a topology other than a tree may cause indirect buffering at other arcs. Before we give a fair accounting we show that the exploitation of this effect is limited but still given for proportional buffering strategies as the UIC's rule.

Definition 3. A buffering is called a *proportional* buffering, if and only if there is some factor $\beta \in \mathbb{Q}^+$, such that $s_a = \beta t_a \ \forall a \in A(G)$.

Definition 4. Construct an unbuffered schedule $\bar{x}$ that respects the technical travel times while minimizing the total (weighted) travel time of passengers assigning to each vertex $v \in V(G)$ a time $\bar{x}_v$. Define the natural slack at arc a as $\ell_a = barx_v - barx_u - t_a$. Then $\sum_{a=(u,v)\in A(G)} \ell_a$ is called the *total natural slack of* G and $\sum_{a=(u,v)\in A(G)} \omega_a \ell_a$ respectively L the *total weighted natural slack of* G.

Theorem 2. Let I_s be the prolongation of the total (weighted) travel time caused by a buffering s to the budget B and L be the total (weighted) natural slack of the network G. If s is proportional with factor β, then $I_s - B = \beta L$.

Proof. Let $\bar{x}$ be the vector of times scheduled to minimize total (weighted) travel time and respecting the technical travel time without buffering. Then $x := (1 + \beta)\bar{x}$ respects the β-proportionally buffered travel times. To show that x minimizes the total (weighted) travel time among all schedules respecting the β-proportional buffering, assume to the contrary that some x' is better. The total (weighted) travel time of x is $1 + \beta$ times that of $\bar{x}$. Now, by β-proportionality of the buffering x' respects we know that $\frac{1}{1+\beta}x' =: \bar{x}'$ respects the technical travel times. Therefore, total (weighted) travel time of $\bar{x}'$ is $\frac{1}{1+\beta}$ of that of x'. This contradicts the minimality of $\bar{x}$.

The indirect buffering in schedule x, $I_s{-}B$, together with the total (weighted) natural slack of G, L, make up for the (weighted) sum over all arcs $a = (u, v)$ of $x_v - x_u - t_a - s_a = (1 + \beta)\ell_a = (1 + \beta)L$. This is the proposition of the theorem. □

Proportional buffers can be understood as scaling of the unbuffered schedule. Thus, the natural slack is scaled proportionally. Note that upper bounds on time durations might conflict with proportional bufferings, as long as they can not be scaled, too. The period time in periodic scheduling is a special case of this. It is not reasonable to scale it. We will exemplify the peculiarities of periodic scheduling in the next section.

In short, the UIC's rule is much less dangerous, than what an optimizer will produce, if he uncarefully derives a budget value from this rule.

The correct, the fair way to account for the price of a certain, delay resistant timetable's costs, is fairly obvious by now. The space of all bufferings is simply $\mathbb{Q}^{|A|}$. We considered the hyperplane of bufferings fulfilling $\sum_{a \in A(T)} s_a = B$, as an optimal buffering will always use the whole budget. Now we give a description of the subspace of fair bufferings. The intersection of the two subspaces will be the domain over which to optimize.

As above we start from an optimal timetable $\bar{x}$ for a graph without buffers. For every arc $a = (u, v) \in A(G)$ the time difference can be written as $x_v - x_u = t_a + \ell_a$, where ℓ_a is the natural slack at arc a and $L = \sum \ell_a$. Observe that the schedule x is tight for the arc length $(t_a + \ell_a)$. A fair buffering is described by a function $\sigma : V(G) \to \mathbb{Q}$ that moves the scheduled time of each vertex in time. The resulting buffering of an arc a is $s_a = x_v + \sigma_v - x_u - \sigma_u - t_a - \ell_a$. In other words, a buffering s_a is fair, if and only if it gives rise to a tight schedule, i.e., for every arc a holds $t_a + \ell_a + s_a = x_v - x_u$. With this definition the following statements are true:

1. The objective $E[\omega^\mathsf{T} d^r]$ is a convex function on the set of fair bufferings.
2. The budget used by a fair buffering equals the increase in passengers' total (weighted) travel time caused by the buffering.

The later being obvious by construction, let us check the first claim. The directed graph G is acyclic and hence allows for a topological ordering. As in the case of trees induction along this order shows that the delay at each vertex v is a convex function: $d_v = \max\{0, \max\{d_u - \ell_a - s_a | a = (u, v) \in A(G)\}\}$.

2.1.3 A Practical, Periodic Example for Implicit Supplements

We illustrate the effect of implicit running time supplements through an example that we derive from practice. The examples feature periodic timetables. They show that in contrast to Theorem 2 in periodic timetables slacks behave disproportional. In the second example the slack even decreases as buffers are raised.

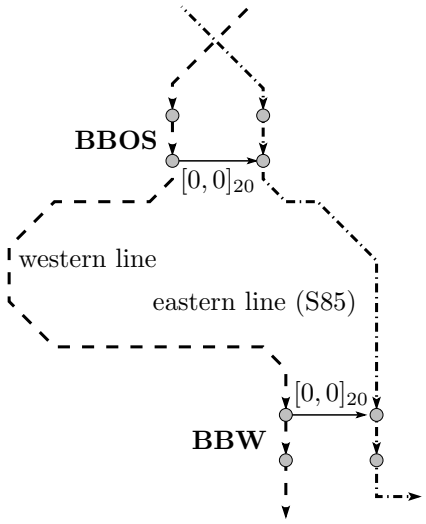


Figure 3: Excerpt of Berlin fast train network (S-Bahn)

Consider the excerpt of the Berlin fast train network (S-Bahn) that we display in Figure 3.

We have to investigate the timetables of two hypothetical — though not unrealistic — lines which meet in two stations, both located on the border of the 'Berlin Ring' around the city center. Both lines are operated with a period time of $T = 20$ minutes. Due to important transfer relations within these two stations it is a marketing requirement that the trains of these two lines have to meet each other in *both* stations. Since this additional requirement prevents the network topology from being a tree, it will become the source of implicit running time supplements.

We investigate two families of timetables: $\{A, B, C\}$ and $\{D, E\}$. For the first family, we assume the realistic technical running times of 45 and 25 minutes for the western and eastern line, respectively. As the period time T equals 20 minutes, without any buffering, both meets can be established without involving any implicit running time supplements, or slack time, cf. Timetable A in Table 1. Applying the UIC rule of adding a running time supplement of 7% translates to explicit running time supplements of six minutes in total, hereby causing an implicit buffer of two for the shorter eastern line (Timetable B). In the case of only the explicit running time supplements to count for a buffer budget of the same amount as the UIC supplements, the most delay resistant solutions will shift the complete budget to one single line, hereby causing an implicit buffer of six minutes, too, for the other line, cf. Timetable C. To summarize, only the explicit running time supplements of Timetables B and C equal, while the nominal increase of travel time of the passengers is 12 in Timetable C, compared to only 8 in Timetable B. Of course, we consider this nominal increase

of travel time being more relevant than only focusing on the budget of explicit running time supplements. Hence, in order to prevent such an unfair accounting for a budget of buffer times, we strongly recommend to include implicit running time supplements (slack times), too, in any definition of a budget.

Table 1: Interaction between explicit and implicit running time supplements

	western line				eastern line				network	
Timetable	tech.	expl.	impl.	total	tech.	expl.	impl.	total	expl.	impl.
A	45	–	0	45	25	–	0	25	–	0
B	45	4	0	49	25	2	2	29	6	2
C	45	6	0	51	25	–	6	31	6	6
D	40	–	0	40	11	–	9	20	–	9
E	40	4	8	52	11	1	0	12	5	8
tech.: technical trip time; expl./impl.: explicit/implicit running time suppl.										

The two remaining Timetables D and E illustrate that the implicit running time supplements may even decrease while the explicit running time supplements are raised. Notice that this is obtained by simply applying the UIC rule of buffering. But we profit from the periodicity of the system, i.e., in Timetable E the meet in Station BBW is established between trains different from the ones that join in a meet in Station BBW in Timetable D.

3 Explicit calculations for paths

For tree topologies the above mentioned accounting problems do not figure. The general picture on a tree is the following. The effect of a single buffer of value s_a on an edge $a = (v, w)$ for the objective function—i.e. the expectation of delay—is constituted by two factors, where the term 'factor' may be understood in the mathematical sense. On the one hand, each arc that is a (direct or indirect) successor of a will be protected by s_a in the following sense. Every delay reaching a plus the disturbance occurring on a up to a total amount of s_a will be subtracted from the successors' delays. The effect of a single buffer of value b on arc a reads: $\sum_{e \in S(a)} g_e \cdot \int_0^b xP[d_v = x]dx$, where $S(a)$ is the set of successors of a. In other words, a buffer has to find the balance between protecting many arcs, as early buffers do, and catching disturbances from many arcs, which is likely for late buffers.

We will now solve the following question by a closed expression: Let T be a path with n arcs, the weight function constant $g \equiv 1$, and any disturbance on an arc is either of value δ or 0. Where to put the buffers?

We consider two different distributions both with a buffer budget of exactly δ:

1. exactly one disturbance occurs and the arc where this happens is chosen uniformly at random

2. every arc independently has probability p to experience a disturbance of δ

Proposition 1. Let T be a path with n arcs and assume exactly one disturbance of amount δ occurs on one of the arcs, which is chosen uniformly at random. To minimize the expected delay along the path subject to a fixed buffer budget of δ, the optimum buffer values s_a for the arcs are

$$s_{\lceil \frac{n}{2} \rceil} = \delta, s_k = 0, \forall k \neq \lceil \frac{n}{2} \rceil. \tag{1}$$

Proof. Let x^r be the position of the disturbance in scenario r. The buffer s_k will be useful if and only if $x^r \leq k$, and the buffer will protect his and any of the following arcs. Thus the effect of s_k on the expectation is $-(n-k+1)P(x \leq k)s_k$. As the budget equals the total delay, the formula is true for every fraction of the buffer budget independent of the position of the rest. Therefore a value k for which this expression is minimal will be an optimal position for the *whole* buffer. Hence the total effect on the expectation,

$$-\sum_k (n-k+1)\frac{k}{n}s_k \tag{2}$$

is optimal for $s_{\lceil \frac{n}{2} \rceil} = \delta$ and $s_k = 0, \forall k \neq \lceil \frac{n}{2} \rceil$. □

Proposition 2. Let T be a path with n arcs and assume that on each arc a delay of δ occurs independently with probability p. To minimize the expected delay along the path subject to a fixed buffer budget of δ, the an optimum buffering is obtained by accumulating the complete budget δ at one arc with index i, such that i maximizes

$$f(i) = (1-(1-p)^i)(n-i+1). \tag{3}$$

Proof. The buffer is again accumulated, because the problem can be formulated as an integer program with a totally unimodular matrix.

$$\begin{aligned}
\text{minimize} \quad & \sum_{r \in R} p^r \sum_{i=0}^{n} d_i^r \\
\text{s.t.} \quad \sum_{i=1}^{n} s_i &\leq B \\
\delta_1^r - s_1 &\leq d_1^r \\
d_{i-1}^r + \delta_i^r - s_i &\leq d_i^r \quad \forall 1 < i \leq n \\
d_i^r &\geq 0
\end{aligned}$$

The budget B being an integer multiple of δ the program can be divided by δ. The total unimodularity then follows from the consecutive ones property.

If a delay occurs on or before the arc i—which has probability $(1-(1-p)^i)$—a buffer that has been placed right here protects $n-i+1$ arcs. □

Let us provide an immediate interpretation of the Function $f(i)$ in Equation 3. On the one hand, buffers tend to be early, because the effect of any delay they catch will be annihilated for the (long) rest of the path. On the other hand, being too early risks that it is set in vain, because the probability $(1-(1-p)^i)$ to catch a sufficiently big delay is not high enough.

This results in a pattern that supports the experimental observations of Kroon et al. [4]. In particular, in the setting of Proposition 2 that is closest to the empirical study of Kroon et al., i.e. $n = 10$ and $p = \frac{1}{5}$, the buffer should be placed on a trip that is strictly before the mid-position of the path (the fourth trip), which coincides with the conclusions of their study, too. For $n = 10$ and $p = \frac{1}{10}$ the fifth trip is the optimal position of the buffer, but the exact, fractional value for i maximizing the effect function $f(i)$ is ~ 4.78. Denote by $g(p, n)$ the function such that $f(g(p, n))$ is maximum. Notice that $g(p, n)$ involves the polylogarithm function. See Figure 4 for a plot of the function $g(p, n)$. Observe that even for a buffer budget proportional to the expected total disturbance, i.e., $p = \frac{1}{n}$, but sufficiently large n the optimal buffer is not in mid-position. For instance, with $n = 20, p = \frac{1}{20}$ the optimal position is 9.

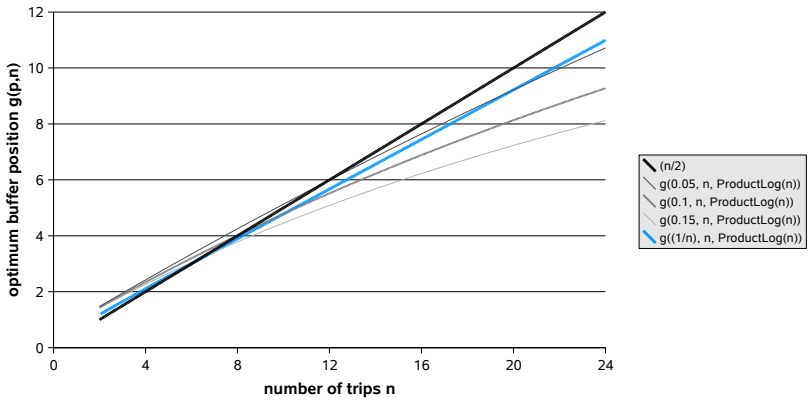


Figure 4: The position $g(p, n)$ maximizing $f(i)$, in function of the path length n

Especially, the first setting could appear academic at first sight. Now, extend the setting to arbitrary p and budgets of some integer multiple of δ, the standard length of a disturbance. This can be formulated as an integer program by scenario expansion. For small n IP-solvers can tackle these problems quickly. If p is small enough all scenarios with more than one disturbance become immaterial to the optimal solution. Those without disturbance anyway have no contribution to the expected delay. Thus the optimal buffer is determined by the cases where exactly one disturbances occurs, which is exactly the first setting. More precisely, because of the integrality of the optimal solution it can be shown that for suitably small p any aberration from the optimal solution to the scenarios with exactly one disturbance weighs heavier than what can in expectation be achieved in the other scenarios.

Figure 5 shows optimal bufferings for $n = 9$, and different values of p, and budgets $0 \leq B \leq 9$ represented in the rows. The columns of each table stand for the 9 arcs of the path. It can be observed how the buffers tend to be earlier, as the probability for disturbances grows.

As we allow for negative buffers the optimization uses these as soon as the budget is unreasonably small. It can be shown that only the last buffer is negative. It is also possible to coin the expression 'unreasonably small' into exact mathematical terms. But this requires some further considerations beyond the scope of this paper.

4 Optimization Methods and Complexity

Kroon et. al [4] proof that a Sample Average Approximation (SAA) approach to optimize the allocation of a fixed budget of supplements over a path with two terminals converges. For arbitrary directed acyclic graphs they propose similar methods without proof. This can be proven by the following consideration:

Calculating the expectation of the minimal makespan for a directed acyclic precedence graph with stochastic arc lengths is an NP-hard problem already for simple distributions (it even reduces from a $\#P$-hard problem [3]) . The idea of the SAA approach is to calculate the average minimal makespan for a random sample of the scenarios only. It is know that the method convergence rapidly for the expected makespan on directed acyclic graphs [8]. Our problem can be expressed by the same notions: Introduce extra arcs each with a *fixed* length from one initial vertex to all other vertices (cf. Figure 6). Thereby we ensure the nominal schedule to be respected. Moreover our objective function is a weighted sum of makespans of each vertex as terminal.

Therefore, if the optimal solution of the makespan (for any vertex as terminal) calculated for the sample is an α approximation for the expected makespan of the entire set of scenarios, then so is the optimal buffering for the scenarios of the random sample a β approximation of the optimal buffering for the expected delay over all scenarios, where α and β differ by some multiplicative constant.

To put it formal, let $c^N(f)$ be the average delay over the sample set N for buffering f, and $c(f)$ the corresponding expectation over the entire set of scenarios. The optimal buffering for the entire set of scenarios is denoted by f^*, and the optimal buffering for the sample N (i.e., the buffering we can compute) is f^N. We have: $|c(f^N) - c^N(f^N)| \leq \beta$ and $|c(f^N) - c^N(f^N)| \leq \beta$ by the approximation. Optimality of f^* and f^N give $c^N(f^N) \leq c^N(f^*)$ and $c(f^*) \leq c(f^N)$. Distinguish three cases: First, assume $c(f^N) \leq c^N(f^N)$. Then we can combine the optimality statements to $c(f^*) \leq c(f^N) \leq c^N(f^N) \leq c^N(f^*) \leq c(f^*) + \beta$, where the last inequality stems from the approximation guarantee. Analogously we conclude in case $c^N(f^*) \leq c(f^*)$ $c^N(f^N) \leq c(f^*) \leq c^N(f^N) + \beta$. It remains to tackle the case $c^N(f^N) \leq c(f^N) \wedge c(f^*) \leq c^N(f^*)$. Then the approximation reads: $c^N(f^*) - c(f^*) \leq \beta$ and $c(f^N) - c^N(f^N) \leq \beta$. By the latter and optimality of f^N we get $c(f^N) - \beta \leq c^N(f^N) \leq c^N(f^*)$. Substituting in the other approximation equality gives $c(f^N) - c(f^*) \leq 2\beta$. By the same

	1	2	3	4	5	6	7	8	9
0									
1					1				
2			1			1			
3			1		1		1		
4		1		1		1		1	
5		1	1		1	1		1	
6		1	1	1		1	1	1	
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(a) $p = 0.04$

	1	2	3	4	5	6	7	8	9
0					1				-1
1					1				
2			1			1			
3			1		1		1		
4		1		1		1		1	
5		1	1		1	1		1	
6		1	1	1		1	1	1	
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(b) $p = 0.05$

	1	2	3	4	5	6	7	8	9
0				1					-1
1				1					
2			1			1			
3			1		1				
4		1		1		1		1	
5		1	1		1	1		1	
6		1	1	1		1	1	1	
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(c) $p = 0.1$

	1	2	3	4	5	6	7	8	9
0				1			1		-2
1				1			1		-1
2			1				1		
3			1		1				
4		1		1		1		1	
5		1	1		1	1		1	
6		1	1	1		1	1	1	
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(d) $p = 0.15$

	1	2	3	4	5	6	7	8	9
0			1		1		1		-3
1			1		1		1		-2
2			1		1		1		-1
3			1		1		1		
4		1		1		1		1	
5		1	1		1	1		1	
6		1	1	1		1	1	1	
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(e) $0.2 \leq p \leq 0.25$

	1	2	3	4	5	6	7	8	9
0		1		1	1		1		-4
1		1		1	1		1		-3
2		1		1	1		1		-2
3		1		1	1		1		-1
4		1		1	1		1		
5		1	1		1	1		1	
6		1	1	1		1	1	1	
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(f) $p = 0.3$

	1	2	3	4	5	6	7	8	9
0		1	1		1	1		1	-5
1		1	1		1	1		1	-4
2		1	1		1	1		1	-3
3		1	1		1	1		1	-2
4		1	1		1	1		1	-1
5		1	1		1	1		1	
6		1	1	1	1		1	1	
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(g) $p = 0.35$

	1	2	3	4	5	6	7	8	9
0		1	1	1		1	1		-5
1		1	1	1		1	1		-4
2		1	1	1		1	1		-3
3		1	1	1		1	1		-2
4		1	1	1		1	1		-1
5		1	1	1		1	1		
6		1	1	1	1		1	1	
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(h) $p = 0.4$

	1	2	3	4	5	6	7	8	9
0		1	1	1	1		1		-6
1		1	1	1	1		1		-5
2		1	1	1	1		1		-4
3		1	1	1	1		1		-3
4		1	1	1	1		1		-2
5		1	1	1	1		1		-1
6		1	1	1	1				
7		1	1	1	1	1	1	1	
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(i) $p = 0.45$

	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1	-8
1	1	1	1	1	1	1	1	1	-7
2	1	1	1	1	1	1	1	1	-6
3	1	1	1	1	1	1	1	1	-5
4	1	1	1	1	1	1	1	1	-4
5	1	1	1	1	1	1	1	1	-3
6	1	1	1	1	1	1	1	1	-2
7	1	1	1	1	1	1	1	1	-1
8	1	1	1	1	1	1	1	1	
9	1	1	1	1	1	1	1	1	1

(j) $0.5 \leq p \leq 1$

Figure 5: Optimal buffer vectors for standard disturbances

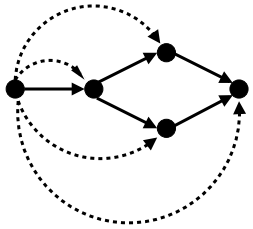


Figure 6: Dashed arcs ensure the nominal timetable

token we get $c(f^*) - c(f^N) \leq 2\beta$ which completes the proof.

The problem about these methods are summarized in [9]. Here the bottom line, adjusted to our problem is this. The sampling approach with sample length N β-approximates the optimal buffering with a probability ξ. Thereby, the probability ξ grows exponentially with the sample size N. But the number of required samples N is not necessarily polynomial in β.

Our approach rests on calculating the expectation of the makespan, which is NP-hard on general directed acyclic graphs even for simple distributions. But this problem is solvable in pseudo-polynomial time for series-parallel graphs. Computational experiments have shown that even for non-series-parallel directed acyclic graphs one gets quite a good heuristic from a certain procedure of series-parallelizing the directed acyclic graph(cf. [7]).

Can we carry over these results to the buffering problem? At first sight this seems problematic, because the extra arcs which we insert to ensure the nominal schedule to be respected in every scenario's dispatching schedule, i.e., the modeling of the buffers, destroy series-parallelity. The example for our procedure in Figure 6 features a series-parallel graph, which is no longer series-parallel after the insertion.

Nevertheless, the expectation in such a graph can be calculated in pseudo-polynomial time. The crucial step is to calculate the distribution of the earliest start time at every vertex one after the other by a simple operations, namely summing the distributions or taking their maximum. This is possible in a series-parallel graph if one follows the serial (summing) and parallel (maximizing) construction of the graph. We can model the respect for the nominal, buffered schedule differently and, not inserting the arcs, preserve series-parallelity: Calculate the vertices' distribution following the series-parallel structure. Every time for a vertex v a distribution has been calculated, add a further maximizing operation with the constant corresponding time of the nominal schedule x_v. Therefore, it is possible to solve the buffering problem (exactly, and with probability 1) for series-parallel graphs.

5 Heuristics for periodic timetabling

5.1 Absorbing path

A PESP solver minimizes the waiting times, which are in some sense the natural defenses of a schedule against delays. In this section we explain a model to impose a certain level of delay resistance defined by absorbing paths on the solution. The idea of an absorbing path is simple. A limited disturbance that occurs at the first arc of the path shall be absorbed at least by the end of the path—assuming no further disturbances.

This approach easily fits into a MIP formulation. Let $P = (a_1, a_2, \ldots, a_k)$ be an absorbing path given as a tuple of edges. First, we restrict ourselves to directed paths. For each arc $a_i = (v_i, v_{i+1})$ the PESP instance specifies lower and upper bounds, ℓ_{a_i} and u_{a_i}. A disturbance of size at most δ at a_1 can be absorbed until v_k, if and only if the slacks between actually chosen potential differences and the lower bounds sum up along the path to a value of at least δ. We enforce slack by adding supplements to the lower bounds. Then the lower bound conditions in the mixed integer formulation of the PESP with variable $\xi_a \geq 0$ change to:

$$\ell_a + \xi_a \leq x_v - x_w + p_a T \tag{4}$$

This allows the following requirement for every absorbing path P:

$$\sum_{a_i \in P} \xi_{a_i} \geq \delta \tag{5}$$

This can be extended to arbitrary paths by introducing corresponding variables $\zeta_a \geq 0$ for the upper bounds:

$$x_v - x_w + p_a T \leq u_a - \zeta_a \tag{6}$$

Then the condition for an (not necessarily directed) absorbing path P will read:

$$\sum_{a_i \in P} \xi_{a_i} + \sum_{-a_i \in P} \zeta_{a_i} \geq \delta \tag{7}$$

In this way the mixed integer program formulation requires no further integer variables. The additional linear inequalities do not delay the solver substantially.

5.2 Periodic timetabling for a strict no-wait policy

Throughout the paper we assumed the delay occurring on an edge to spill over to its ancestors. In real-world systems dispatching decisions may neutralize this effect. In other words, the dispatcher may choose to set a precedence constraint out of force. Optimal dispatching decisions are hard problems in their own

right [2]. Our assumption, that every delay spills over, corresponds to an all-wait policy of the dispatcher. In this section we will assume to the contrary a strict no-wait policy.

As an example think of a city's subway system operated with a periodic timetable of periodicity T. If the train of one line is delayed the trains of the connecting lines usually will not wait for the passengers of the late train. Thus, these passengers experience a waiting time of almost T. More precisely, if their train is delayed by d, and the connection was tight up to the time necessary to walk to the other platform, they have to wait $T - d$ for their connection. The model is accurate if only connection arcs are considered. But any arcs expressing infrastructural constraints cannot be set out of force by any dispatcher. A train, as a physical object, that runs from A via B to C cannot start its trip to C before it reached B. Here an all-wait policy is a matter of fact.

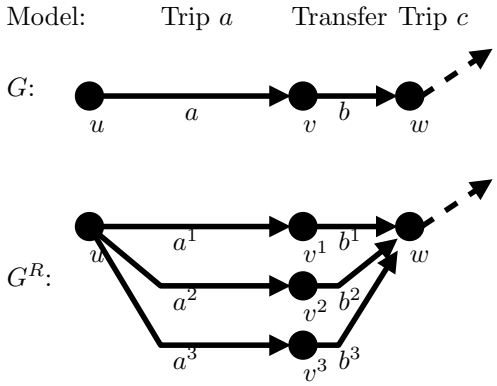
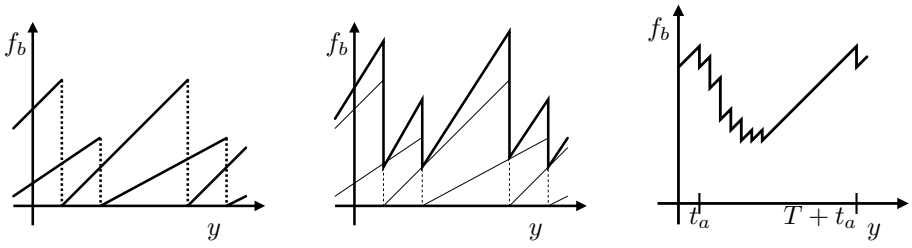


Figure 7: The scenario expansion

To understand the behavior of the system for a strict no-wait policy we introduce a scenario expansion G^R of the network. The stochastic model consists—as before—of an independent distribution for finitely many states $\{t_a^1, \ldots, t_a^{k_a}\}$ on each trip a. (In fact, the independence will be immaterial in our model.) Substitute the vertex v of the arrival event of a trip a by k_a-many vertices $\{v^1, \ldots, v^{k_a}\}$ and connect u, the start vertex of a, with each of them. The length of arc $a^i = (u, v^i)$ is t_a^i . Then connect every vertex v^i to all vertices that v was connected to. Let b^i be such an arc from vertex v^i and b its paradigm edge. Then the weight (passenger load) of b^i shall be the weight of b times the probability p^i that trip a takes t_a^i time units. This is in expectation the passenger load that will change to w, the end vertex of b, after having traveled t_a^i time units to make the trip a.

What will happen to the new network G^R when we apply a solver creating timetables with minimal total, weighted waiting time? The trip length t_a^i in the scenarios are fixed. Therefore our means to optimize rest in adjusting the difference $x_w - x_u$. It may well be that this difference chosen to be smaller than t_a^i for some scenarios. Consequently, $x_w - x_{v^i}$ will be negative, say $x_w - x_{v^i} = -\Delta$.



(a) Two different trip length t_a^1, t_a^2 with probabilities $\frac{1}{3}$ and $\frac{2}{3}$

(b) The sum of the two objective functions

(c) Sum over more than two trip lengths

Figure 8: Piecewise linear objective functions, $y = x_w - x_u$

In the periodic setting this means that the edge b^i will count in the objective with $T - \Delta$. This is precisely, what we want the model to do. If the train to which b is the connecting arc starts according to the schedule, the passengers that experience a trip length of t_a^i for trip a and miss their connection, will have to wait for the next train $T - \Delta$ time units.

In general we define $d_b^i(x) = x_w - x_u - t_a^i - t_b \mid \mod T$ as the contribution of arc b to the objective function in scenario i under schedule x. Summing up the contributions of each trip length we get a function that grows linearly but drops whenever a trip length equals the argument value modulo T. Precisely speaking, we have for the contribution f_b of edge b to the objective function by the choice of $x_w - x_u$:

$$f_b(x_w - x_u) = \sum_i p^i(d_b^i(x)) \tag{8}$$

The function d^i increases linearly, in fact with factor 1 at every point except for those points, where a multiple of T is reached. We can reformulate

$$f_b(y) = y - \sum_{i=1}^{k_a} p^i \sum_{z=0}^{y} \chi_i(z) + C, \tag{9}$$

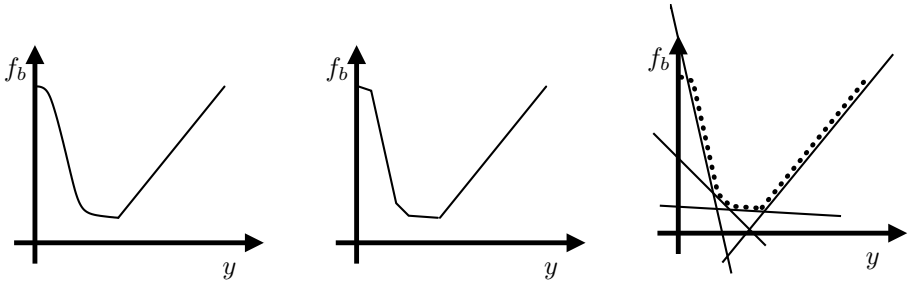
where $\chi_i(y)$ is 1 if $\exists \ell \in \mathbb{Z} : y = \ell T - t_a^i - t_b$ and 0 else. The constant C is due to the minimal travel time on trip a and the minimum transfer time to trip c. As a constant it is irrelevant in optimization.

In Figure 5.2 we see the functions corresponding to two groups passengers. The bigger group travels t_a^1 on trip a, the smaller one t_a^2. The size, i.e., the probability p^i for any passenger to be in one of these groups is reflected in the slope of their functions. The moment they could catch the train for trip c without any waiting time their contribution drops to zero, indicated by the dotted line. The sum of both functions—assuming that these comprise all passengers—has slope equal to 1 (Figure 5.2). The beginning of the big straight line in Figure 5.2 indicates, that by that time any passenger managed to reach the door of the train for trip c.

Assume we are given a measurable distribution $\mu : \mathbb{Q} \to [0,1]$ giving for every trip length t its probability $\mu(t)$ (at arc a). Then we can take limits in Equation (9) and

$$f_b(y) = \int_0^y (1 - \mu(t))dt + C'. \tag{10}$$

For instance, an exponential distribution of the travel times e^{-t} gives $f(y) = y - e^{-t} + C$. For computational purposes we approximate such a function by a piecewise linear function. In solving a PESP, convex piecewise linear objective functions can be handled well by adding linear constraints (confer Figure 9). The convexity of this function is not guaranteed a priori, but a reasonable assumption.



(a) The original objective function

(b) A piecewise linear approximation

(c) Modeling by linear constraints that enforce convexity

Figure 9: Linearizing the objective

In fact the idea to use such convex, piecewise linear function is not surprising. Minimizing waiting times is a typical objective for periodic timetabling. Such tight schedules are not likely to be delay resistant. Therefore, it seems reasonable to punish both, very long and very short connections. By the considerations we have a better understanding, what the objective function should look like, in particular as Equation (10) gives a direct connection to the distribution of the trip length. Moreover we better understand the problems of this approach: First, as already mentioned, the model can only describe no-wait edges. Secondly, the model is over pessimistic. Sometimes you are so lucky that although your train is late you get the connection because the second train is late, too. As our model allows for no spill over effect it also does not take such a compensation into account. In case of bad weather many trains can be a little late, and thus compensation effects become likely. The model is incapable of modeling the dependencies between the distributions.

References

[1] D. Bertsimas and M. Sim (2003) Robust Discrete Optimization and Network Flows. Mathematical Programming Series B **98**, 49–71

[2] M. Gatto, R. Jacob, L.W.P. Peeters, and A. Schöbel (2005) The computational complexity of delay management. In D. Kratsch: Graph-Theoretic Concepts in Computer Science (WG 2005), 227–238, Springer LNCS 3787

[3] J. Hagstrom and N. Jane (1990) Computing the probability distribution of project duration in a PERT network. Networks **20**, 231–244

[4] L.G. Kroon, R. Dekker, and M. Vromans (2005) Cyclic Railway Timetabling: A Stochastic Optimization Approach. To appear in Springer LNCS Volume on Algorithmic Methods for Railway Optimization, Preprint available at `http://www.few.eur.nl/few/research/ecopt/publications`

[5] C. Liebchen, M. Proksch and F. Wagner (2004) Performance of algorithms for periodic timetable optimization. In M. Hickman (Ed.): Proceedings of the Ninth International Workshop on Computer-Aided Scheduling of Public Transport (CASPT), to appear

[6] C. Liebchen and R.H. Möhring (2004) The modeling power of the periodic event scheduling problem: Railway timetables – and beyond. In M. Hickman (Ed.): Proceedings of the Ninth International Workshop on Computer-Aided Scheduling of Public Transport (CASPT), to appear

[7] A. Ludwig, R. Möhring and F. Stork (2001) A computational study on bounding the makespan distribution in stochastic project networks. Annals of Operations Research **102**, 49–64

[8] R. Van Slyke (1963) Monte Carlo methods and the PERT problem. Operations Research **11**, 839–860

[9] A. Shapiro and A. Nemirovski (2005) On complexity of stochastic programming problems. In V. Jeyakumar and A.M. Rubinov (Eds.): Continuous Optimization: Current Trends and Applications, 111-144 , Springer

[10] U.I.C. (2000) Timetable recovery margins to guarantee timekeeping — Recovery margins, Leaflet 451-1, Union Internationale des chemins de fer, Paris, France

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

2006/24 *Christian Liebchen and Sebastian Stiller:* Delay Resistant Timetabling

2006/08 *Nicole Megow and Tjark Vredeveld:* Approximation Results for Preemptive Stochastic Online Scheduling

2006/07 *Ekkehard Köhler and Christian Liebchen and Romeo Rizzi and Gregor Wünsch:* Reducing the Optimality Gap of Strictly Fundamental Cycle Bases in Planar Grids

2006/05 *Georg Baier and Thomas Erlebach and Alexander Hall and Ekkehard Köhler and Heiko Schilling:* Length-Bounded Cuts and Flows

2005/30 *Ronald Koch and Martin Skutella and Ines Spenke :* Maximum k-Splittable Flows

2005/29 *Ronald Koch and Ines Spenke :* Complexity and Approximability of k-Splittable Flows

2005/28 *Stefan Heinz and Sven O. Krumke and Nicole Megow and Jörg Rambau and Andreas Tuchscherer and Tjark Vredeveld:* The Online Target Date Assignment Problem

2005/18 *Christian Liebchen and Romeo Rizzi:* Classes of Cycle Bases

2005/11 *Rolf H. Möhring and Heiko Schilling and Birk Schütz and Dorothea Wagner and Thomas Willhalm:* Partitioning Graphs to Speed Up Dijkstra's Algorithm.

2005/07 *Gabriele Di Stefano and Stefan Krause and Marco E. Lübbecke and Uwe T.Zimmermann:* On Minimum Monotone and Unimodal Partitions of Permutations

2005/06 *Christian Liebchen:* A Cut-based Heuristic to Produce Almost Feasible Periodic Railway Timetables

2005/03 *Nicole Megow, Marc Uetz, and Tjark Vredeveld:* Models and Algorithms for Stochastic Online Scheduling

2004/37 *Laura Heinrich-Litan and Marco E. Lübbecke:* Rectangle Covers Revisited Computationally

2004/35 *Alex Hall and Heiko Schilling:* Flows over Time: Towards a more Realistic and Computationally Tractable Model

2004/31 *Christian Liebchen and Romeo Rizzi:* A Greedy Approach to Compute a Minimum Cycle Bases of a Directed Graph

2004/27 *Ekkehard Köhler and Rolf H. Möhring and Gregor Wünsch:* Minimizing Total Delay in Fixed-Time Controlled Traffic Networks

2004/26 *Rolf H. Möhring and Ekkehard Köhler and Ewgenij Gawrilow and Björn Stenzel:* Conflict-free Real-time AGV Routing

2004/21 *Christian Liebchen and Mark Proksch and Frank H. Wagner:* Performance of Algorithms for Periodic Timetable Optimization

2004/20 *Christian Liebchen and Rolf H. Möhring:* The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables — and Beyond

2004/19 *Ronald Koch and Ines Spenke:* Complexity and Approximability of k-splittable flow problems

2004/18 *Nicole Megow, Marc Uetz, and Tjark Vredeveld:* Stochastic Online Scheduling on Parallel Machines

2004/09 *Marco E. Lübbecke and Uwe T. Zimmermann:* Shunting Minimal Rail Car Allocation

2004/08 *Marco E. Lübbecke and Jacques Desrosiers:* Selected Topics in Column Generation

2003/050 *Berit Johannes:* On the Complexity of Scheduling Unit-Time Jobs with OR-Precedence Constraints

2003/49 *Christian Liebchen and Rolf H. Möhring:* Information on MIPLIB's timetab-instances

2003/48 *Jacques Desrosiers and Marco E. Lübbecke:* A Primer in Column Generation

2003/47 *Thomas Erlebach, Vanessa Kääb, and Rolf H. Möhring:* Scheduling AND/OR-Networks on Identical Parallel Machines

2003/43 *Michael R. Bussieck, Thomas Lindner, and Marco E. Lübbecke:* A Fast Algorithm for Near Cost Optimal Line Plans

2003/42 *Marco E. Lübbecke:* Dual Variable Based Fathoming in Dynamic Programs for Column Generation

2003/37 *Sándor P. Fekete, Marco E. Lübbecke, and Henk Meijer:* Minimizing the Stabbing Number of Matchings, Trees, and Triangulations

2003/25 *Daniel Villeneuve, Jacques Desrosiers, Marco E. Lübbecke, and François Soumis:* On Compact Formulations for Integer Programs Solved by Column Generation

2003/24 *Alex Hall, Katharina Langkau, and Martin Skutella:* An FPTAS for Quickest Multicommodity Flows with Inflow-Dependent Transit Times

2003/23 *Sven O. Krumke, Nicole Megow, and Tjark Vredeveld:* How to Whack Moles

2003/22 *Nicole Megow and Andreas S. Schulz:* Scheduling to Minimize Average Completion Time Revisited: Deterministic On-Line Algorithms

2003/16 *Christian Liebchen:* Symmetry for Periodic Railway Timetables

2003/12 *Christian Liebchen:* Finding Short Integral Cycle Bases for Cyclic Timetabling

762/2002 *Ekkehard Köhler and Katharina Langkau and Martin Skutella:* Time-Expanded Graphs for Flow-Dependent Transit Times

761/2002 *Christian Liebchen and Leon Peeters:* On Cyclic Timetabling and Cycles in Graphs

752/2002 *Ekkehard Köhler and Rolf H. Möhring and Martin Skutella:* Traffic Networks and Flows Over Time

739/2002 *Georg Baier and Ekkehard Köhler and Martin Skutella:* On the k-splittable Flow Problem

736/2002 *Christian Liebchen and Rolf H. Möhring:* A Case Study in Periodic Timetabling

Reports may be requested from:

Sekretariat MA 6–1
Fakultt II – Institut fr Mathematik
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

e-mail: klink@math.TU-Berlin.DE

Reports are also available in various formats from

`http://www.math.tu-berlin.de/coga/publications/techreports/`

and via anonymous ftp as

`ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/Report-`*number*`-`*year*`.ps`